Open Domain Question Answering

KATEŘINA MACKOVÁ

Question Answering

Úloha v oblasti zpracování přirozeného jazyka a získávání informací

Cíl: natrénovat model, který bude umět odpovídat na otázky vztahující se k danému textu

Dříve: pomocí získání klíčových slov z otázky a dedukce odpovědi z databáze znalostí

Dnes: neuronové sítě, které dokáží porozumět významu textu a na základě toho v textu najít odpověď

Open domain Question Answering

Klasický QA model:

- Vstup: text a k němu otázka
- Výstup: odpověď na otázku
- Obvykle je k odpovědi i její počáteční index v textu

Open Domain QA

- Rozšíření klasického QA na open domain datasety jako Wikipedie
- Odpověď zde nemá identifikátor, kde v textu se nachází
  - a tedy může být v kterémkoli z milionu dokumentů
- Proto je potřeba k dané otázce najít potenciálně relevantní dokument a získat z něj odpověď

Jak na to?

Potřebujeme, aby model rozuměl textu, aby byl schopný z něj dedukovat odpovědi na otázky

Nejprve je tedy třeba natrénovat jazykový model

- Ten v sobě bude uchovávat world knowledge a vlastnosti jazyka
- Bude se učit textu porozumět a umět z něj dedukovat smysluplné informace

Potom se může udělat finetuning na úlohu QA

- Zde budou vstupem otázky a odpovědi a model se bude učit na ně odpovídat na základě jazykového modelu, který si natrénoval dříve

Motivace

Pří trénování jazykového modelu je třeba do něj zanést velké množství world knowledge, které je pak klíčové při NLP úlohách (například i QA)

Tyto znalosti jsou ukládány implicitně v parametrech NN

- Proto je potřeba velké NN, aby se tam vše zvládlo uložit – může být problém

Třeba zachytit world knowledge v nějaké rozumnější formě

- Pomocí předtrénování jazykového modelu pomocí knowledge retrieveru
- Ten umožní modelu pracovat nad velkými dokumenty z velkých korpů, například z celé Wikipedie

Tento článek právě ukazuje jak předtrénovat takový knowledge retriever

- Dělá to pomocí unsupervised metod - maskované modelování jazyka a backpropagace

Kvalita vytvořeného modelu se změří finetuningem na úkolu Open-domain QA

- Výsledky pak porovnávají s nejmodernějšími modely pro explicitní i implicitní ukládání znalostí
- Tento model překonává všechny předchozí metody s výraznou odchylkou (4-16% accuracy)
- Zároveň poskytuje kvalitativní přínosy, jako je interpretabilita a modularita

Další přístupy

Další přístupy: BERT

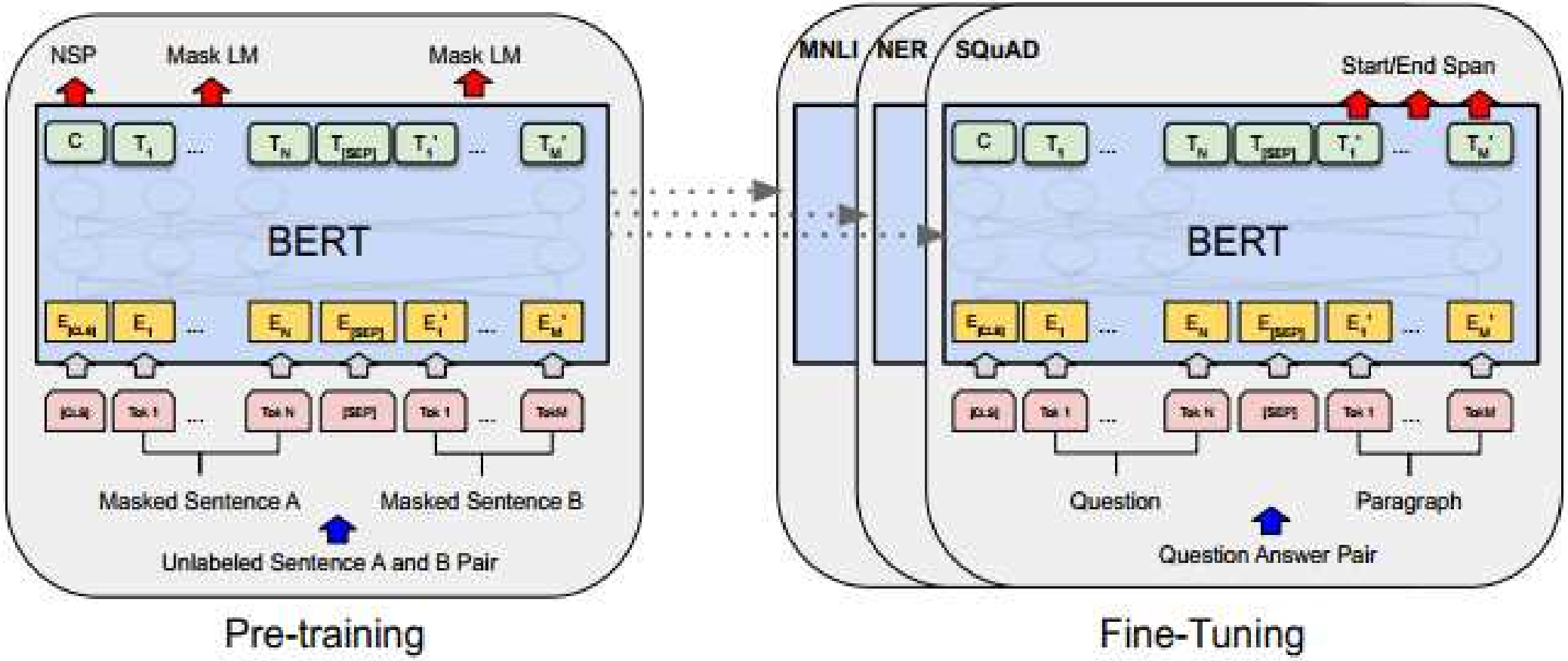
Bidirectional Encoder Representations from Transformers (BERT)

- Jeden z modelů, co používá předtrénování jazykového modelu z neolabelovaných dat

Založený na modelu Transformer

Vytváří se embeddingy slov pomocí levého i pravého okolního kontextu v daném dokumentu pomocí attention mechanismu v trénovaném modelu NN

- Naučené word knowledge jsou zde uloženy v parametrech NN
- Tak se špatně poznává, které znalosti tam jsou uložené a kde přesně
- Navíc je prostor na uložení znalostí limitován velikostí sítě
- Při každém přidání nové znalosti je třeba zvětši síť - pomalé a drahé



- Celková předtrénovací a finetuning procedura pro BERTa
- Kromě výstupní vrstvy se používá stejná architektura
- Předtrénování jazykového modelu – pomocí maskování
- Finnetuning – pomocí QA dvojic

Další přístupy: T5

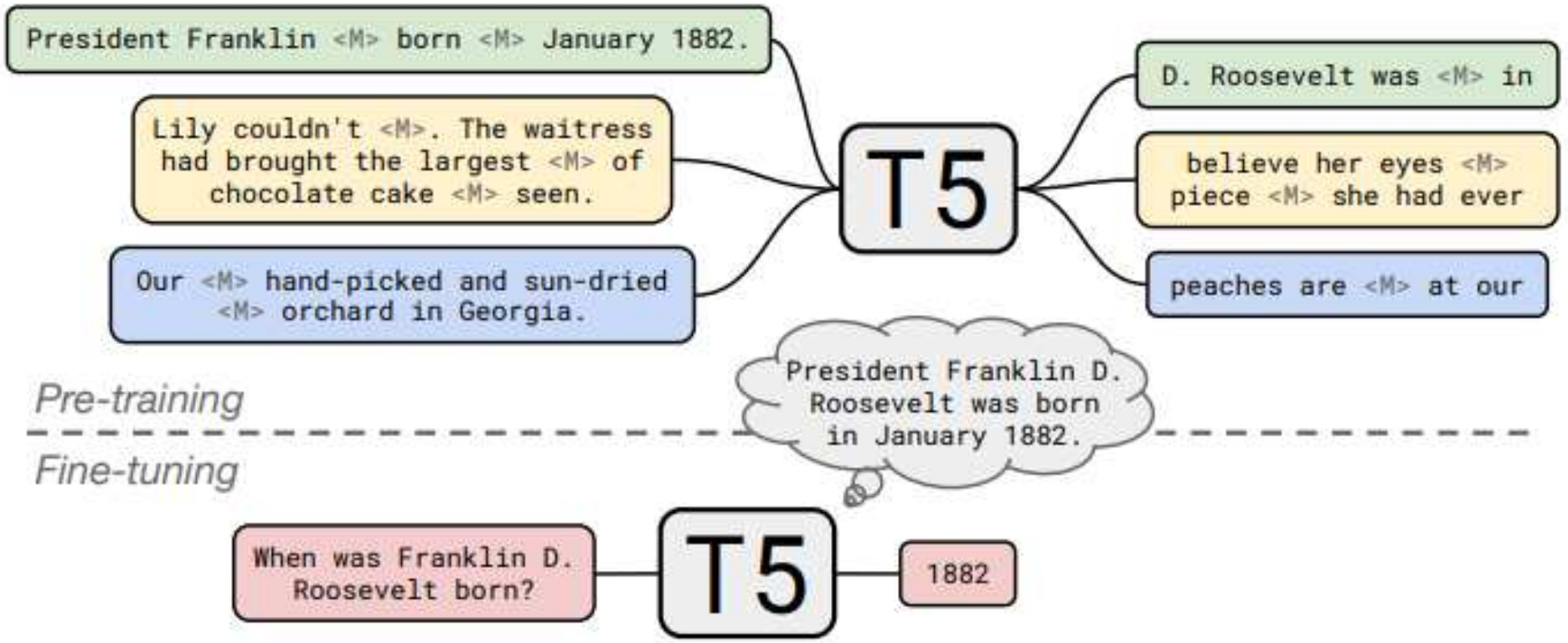
Text-to-Text Transfer Transformer (T5)

Opět vychází z modelu Transformer

- Je upravený tak, aby se používal na širokou škálu úkolů
- Přistupuje k nim jednotně – přijímá vstupní text a vypisuje nějaký výstupní text – typ úlohy je vložen jako popis ke vstupu

Model

- Skládá se z několika bloků, každý s self attention vrstvou a feedforward sítí
- Je nejprve trénován pomocí maskování na unlabeled textu obdobně jako BERT
- Pak finetuning na konkrétních úlohách i s typy úloh



- T5 is předtrénován k doplnění vynechaných slov v textu ve velkém nestrukturovaném korpu
- Pak je finetunován k odpovídání otázek

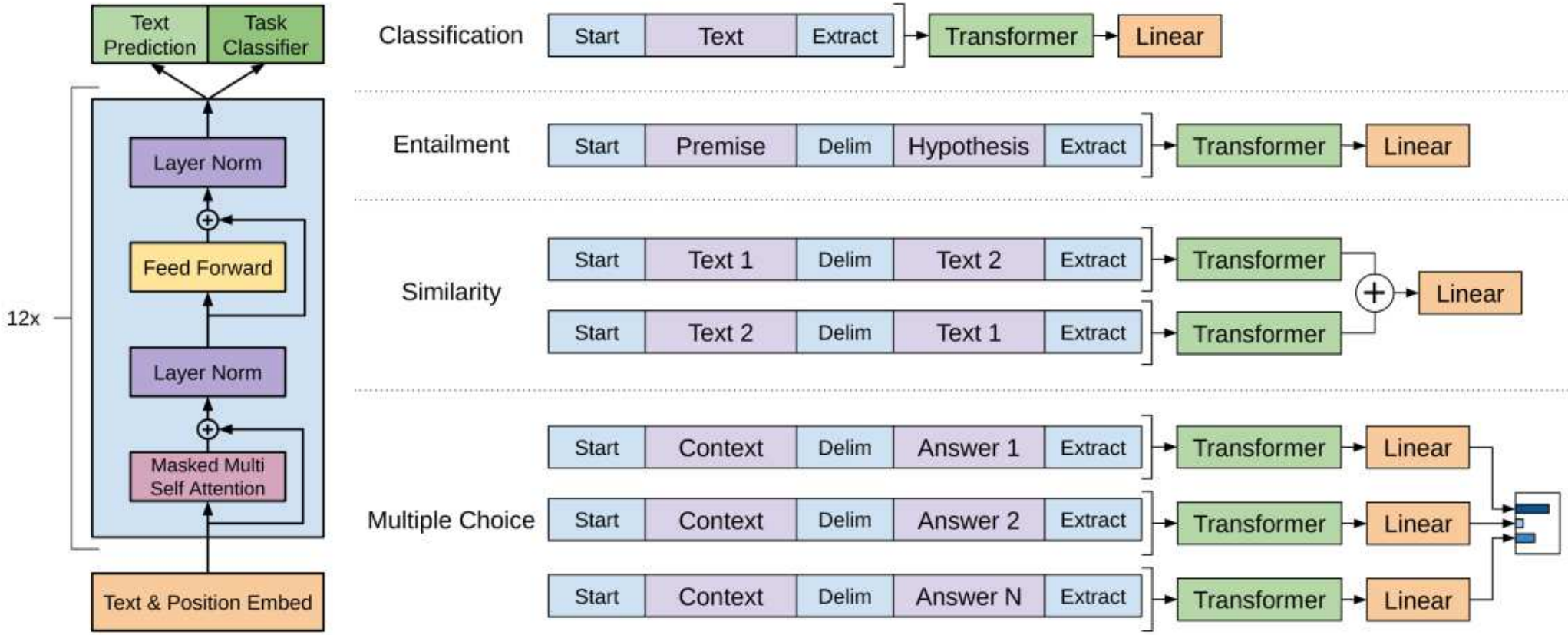
Co je to ten Transformer?

Transformer model je běžný model pro language understanding úlohy

- Původní Transformer se skládá z encoder-decoder architektury a byl určen k sekvence-to-sekvence úkolům
- Tedy trénování modelu ke konverzi sekvencí z jedné domény do druhé
- Hlavně pro strojový překlad, ale i pro další úlohy jako QA

Základní stavební blok je self-attention vrstva

- Ta zpracovává sekvenci nahrazením každého prvku váženým průměrem zbytku sekvence
- Na základě toho se naučí jazykový model
- Pak se využívá pomocí finetuningu k řešení problému dané úlohy



Celková architektura modelu Transformer (vlevo)

Následná transformace na finetuning pro různé úlohy (vpravo)

REALM

Aktuální přístup - REALM

Model REALM (Retrieval-Augmented Language Model Pre-Training)

- K zachycení znalostí v intepretovatelnější formě navrhuje tento model předtrénování s naučeným textual knowledge retrieverem
- Na rozdíl od modelů, co ukládají znalosti do parametrů, tento přístup explicitně nutí model se rozhodnout, které znalosti vytáhnout a využít při odvozování
- Před vytvořením predikce tedy jazykový model použije retriever pro vybrání několika dokumentů z velkého korpu jako Wikipedie
- Pak pracuje nad těmito dokumenty k získání správně předpovědi

REALM – úvod

Skládá se za dvou částí

- Nejprve je třeba natrénovat neural knowledge retriever z unsupervised textu
- jeho zahrnutí do předtrénování je početně náročné, protože retriever musí zvažovat miliony kandidátů v každém kroku
  - Navíc jeho rozhodnutí je pak backpropagováno zpět
  - Užitečná info od bude odměněna a neužitečná penalizovaná
  - Podle toho se bude retriever updatovat
- Proto se používá strukturování retrieveru tak, že výpočty pro každý dokument mohou být updatovány asynchronně, a výběr nejlepšího dokumentu může být formulován jako MIPS (maximum inner product search)
- Potom je model hodnocen finetuningem na úlohu Open QA na 3 benchmarcích:
  - NATURALQUESTIONS-OPEN, WEBQUESTIONS, CURATEDTREC

Porovnávali výsledky hlavně s modelem T5, co ukládá znalosti implicitně a používá k tomu externí paměť

REALM dává lepší výsledky o 4-16% v celkové accuracy

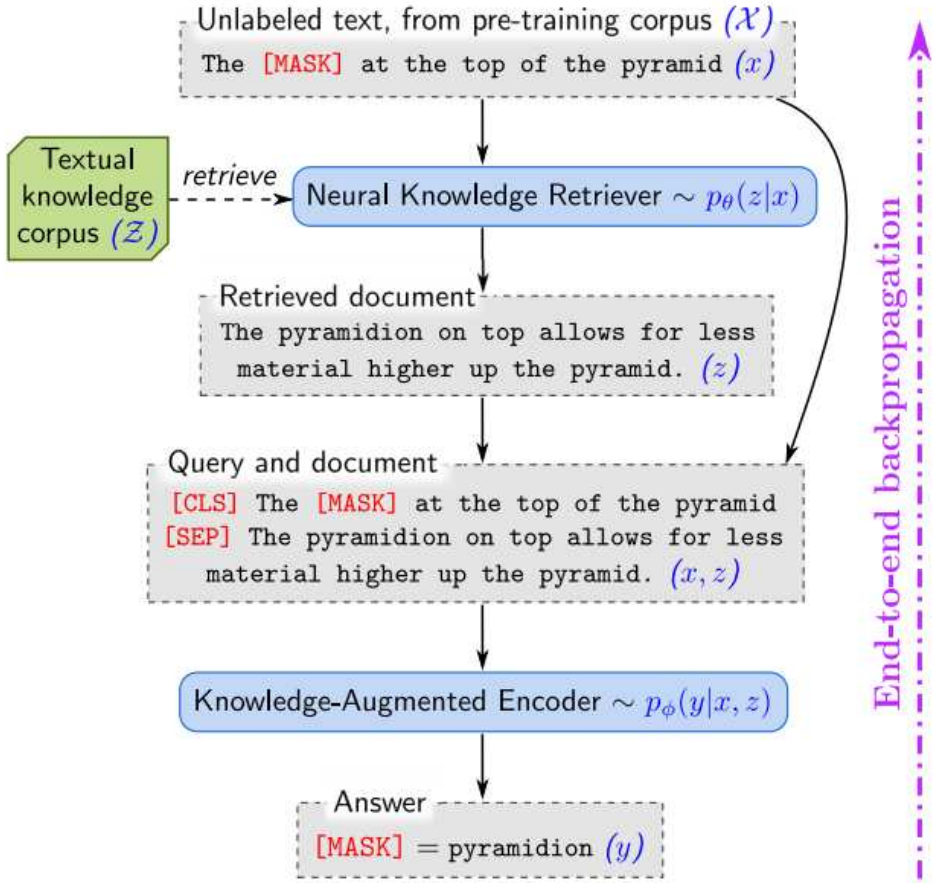
Trénování jazykového modelu

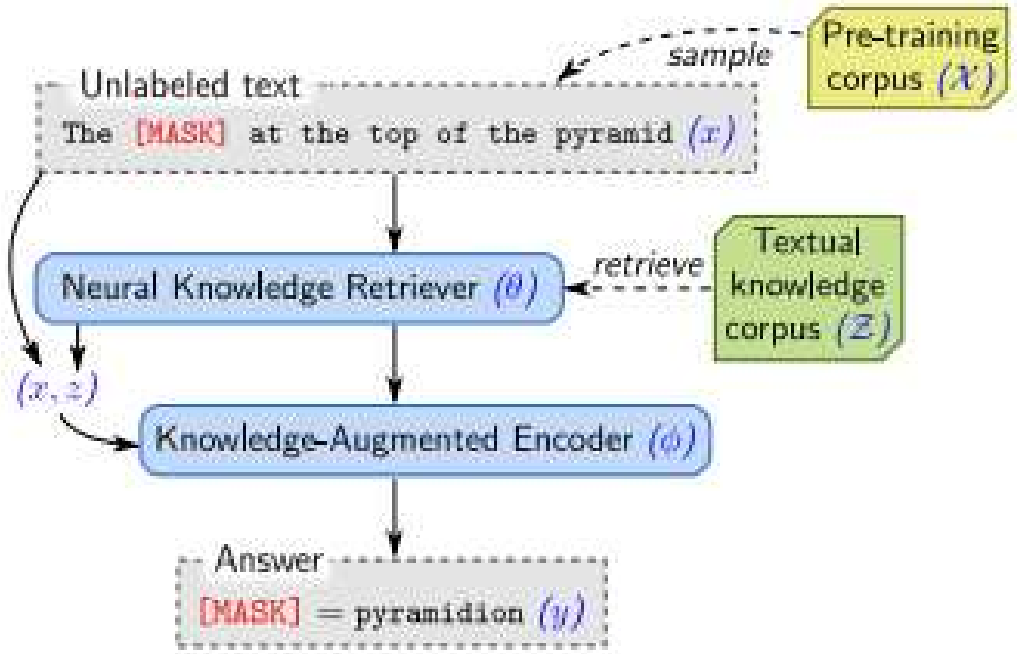
Pomocí neural knowledge retrieveru

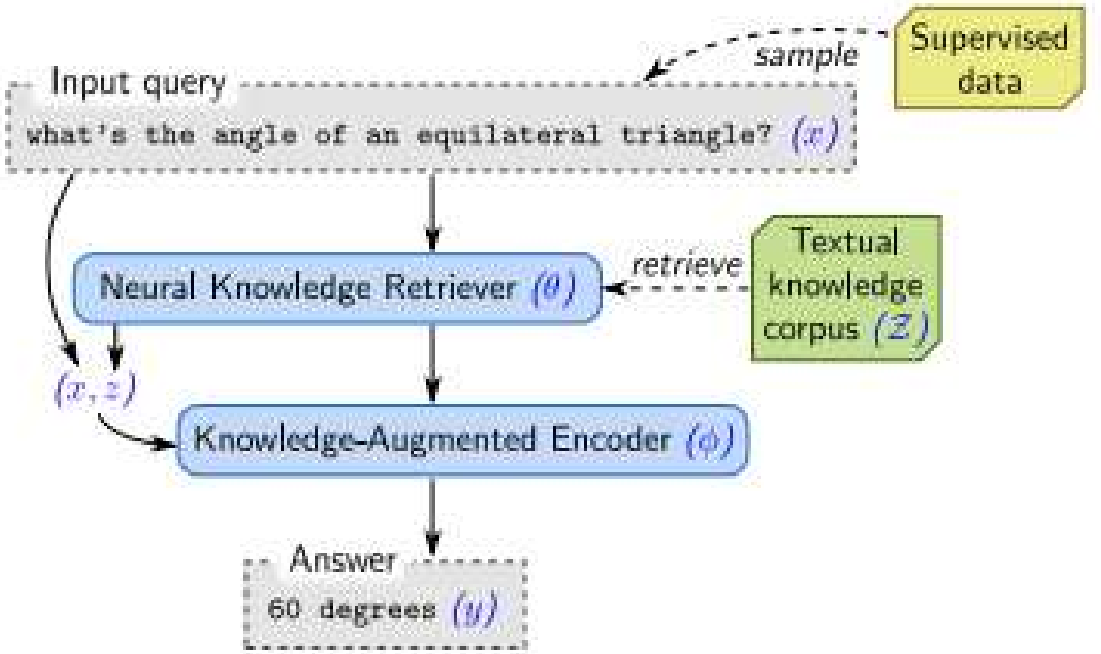
- Získá z celého kontextového korpu dokument pro daný vstup
- Nad ním se učí jazykový model pomocí maskování
- Získaný signál z něj se pak zpětně šíří pomocí backpropagace přes vyhledávač
- Cíl: naučit se užitečnou reprezentaci jazyka z neolabelovaného textu

Výsledný model může být pak finetuningován na konkrétní úlohu NLP

- Obvykle vede k lepším výsledkům než modely bez něj







Model bere vstup x a naučí se pravd. rozdělení p(y|x) nad všemi možnými výstupy y

Vlevo: unsupervised pretraining pomocí maskování:

- X je věta z datového korpu s nějakými odebranými tokeny a model se je snaží doplnit
- Model musí umět zakódovat syntaktickou a sémantickou informaci stejně dobře jako obecné znalosti světa

Vpravo: supervised finetunning na Open QA:

- Po natrénování parametrů retrieveru a encoderu se finetuningují na danou úlohu
- x je otázka a y odpověď

Počítání pravděpodobnosti p(y|x)

Nad daným vstupem x získá retriever možné užitečné dokumenty ze znalostního korpu Z, který modeluje jako pravd. p(z|x)

Pak na základě z a x podmíní y a vygeneruje výstup y jako p(y|z,x)

K získání celkové pravděpodobnosti y označíme z jako skrytou proměnnou a spočítáme

$$p(\text{y}|\text{x}) = \sum_{z \in Z} p(y|z,x) * p(z|x)$$

Máme tedy dvě klíčové komponenty neural knowledge retriever co získává p(z|x) a knowledge-augmented encoder co modeluje p(y|z,x)

Trénovací proces

Trénování i finetuning probíhá pomocí max log pravděpodobnosti p(y|x) správného výstupu y

Následně se z toho spočítá gradient vzhledem k parametrům modelu a optimalizuje se pomocí sochastic gradient descent

Klíčová výpočetní challenge je to, že marginální pravděpodobnost vyžaduje sumy nad všemi dokumenty z v jazykovém korpu Z

$$p(\text{y}|\text{x}) = \sum_{z \in Z} p(y|z,x) * p(z|x)$$

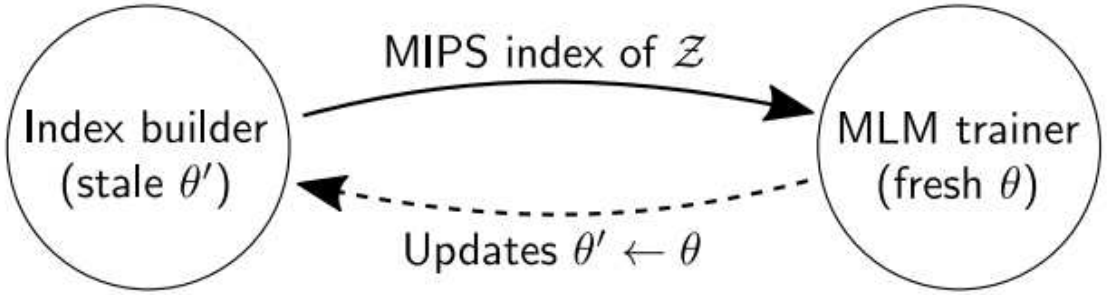
- To je aproximováno sumou nad k dokumenty s nejvyšší pravděpodobností výskytu $p(z|x)$
- Vlastně to ani nevadí, protože většina dokumentů má stejně 0 pravděpodobnost

Trénovací proces

K vypočítání nejlepších k dokumentů se používá asynchronní MIPS algoritmus

- Updatuje MIPS index pomocí běhu 2 paralelních jobů
  - První trénovací job dělá gradientní update na parametru modelu
  - Druhý index builder vybírá a indexuje dokumenty
- Takže se zároveň trénují parametry modelu a builder vytváří indexy na pozadí
  - když najde potenciálně správný index tak ho pošle trénovacímu jobu na update parametrů

Použije se při trainu a při finetuningu už se parametry neupdatují



Další strategie při vývoji REALM

Salient span masking

- Během předtrénování se chceme soustředit na příklady x, které vyžadují znalosti světa k predikci maskovaných tokenů
- MLM (model language modelling) používané u BERTa ale maskuje jen na základě lokálního kontextu
- Proto se používá salient spans, které maskuje entity jako UK nebo July 1969
  - používá se BERT tagger k identifikace jmenných entit a regexpy k identifikaci dat
  - Vybírají se a maskují potom tyto slova, aby se model učil i význam a souvislosti
- Tato technika výrazně překonává jiné maskovací strategie

Null dokument

- Občas nějaké entity nevyžadují znalosti světa k predikci
- proto se přidá do modelu null dokument do nejvyšších k dokumentů
- Používá se, když není potřeba žádné získávání informací k predikci slova

Další strategie při vývoji REALM

Zakázání triviálních retrievals

- Pokud je trénovací korpus X a znalostní korpus Z stejný, existuje triviální kandidát z, který bude nejlepší
  - stačí se jen podívat do z k predikci toho, co tam má být za slovo
- To pak dá velký pozitivní gradient pro p(z|x)
  - při příliš častém opakování by to mohlo skončit hledáním matchování stringů
- proto jsou triviální kandidáti při tréninku vyloučeni

Inicializace

- Na začátku retriever nemá dobré embeddingy pro x a z
- Pak dokumenty získané nebudou relevantní k x
  - Ani gradient se nebude zlepšovat a vytvoří se bludný kruh
- Proto je potřeba inicializovat počáteční embeggindy nějak líp
  - pomocí ICT (Inverse Cloze Task)
    - Máme danou sekvenci a model je trénován k získání dokumentu, odkud pochází

Experimenty: evaluace

Evaluace tedy probíhala na Open QA tasku

Mnoho QA výkonnostních testů bylo navrhnuto tak, že se soustředí na datasety, kde autoři otázek neznali odpovědi

- potom to přináší otázky, které odrážejí realističtější hledání informací
- Nezanáší tam chyby, když je otázka formulovaná s předem známou odpovědí

V každém případě je predikovaná odpověď je vyhodnocena pomocí exact match skore s každou referenční odpovědí

Exact match

- Skóre, kdy model musí odpovědět všechny slova v otázce k získání bodu
- V případě částečně shodné odpovědi nebo špatné odpovědi nezíská nic

Experimenty

Experimenty: datasety

NaturalQuestions-Open

- Skládá se z přirozeně se vyskytujících dotazů Google a jejich odpovědí
- Každá odpověď má ještě svůj typ
- Nechali si pouze odpovědi typu krátké odpovědi s max 5 tokeny
- Dataset poskytuje i originální Wikipedia dokumenty, odkud se dají informace získávat

WebQuestions

- Shromážděn z Google Suggest Api pomocí jedné počáteční otázky a její rozšíření na množinu relevantních otázek

CuradTrec

- Kolekce QA dvojic získaných od uživatelů ze sítí jako s MSNSearch and AskJeeves
- aby bylo možné víc správných odpovědí, tak jsou odpovědi regexpy co matchují všechny správné odpovědi (kvůli chybám ve spellingu a tak)
- Akorát není jasné, jak na tomto trénovat generation-based modely, takže na nich je nepouštěli

Experimenty: přístupy

Retrieval-based Open-QA

- Většina existujících open QA systémů odpoví na vstupní otázku získáním potenciálně relevantního dokumentu z korpu a pak použije reading comprehension systém k získání odpovědi z textu
  - Zde musí být odpověď explicitně uložena v korpu
- Chceme ale porovnat víc metod pro implementaci retrievera
  - Pomocí různých heuristik, co používají entity spojující otázku s výběrem malé množiny relevantních dokumentů:
    - DrQA, HArdEM, GraphRetirever PathReriever
  - Přístupy s MIPS indexem:
    - ORQA, REALM (ten má navíc jazykový model pro předtrénování a backpropaguje index do MIPSU místo použití fixovaného indexu)

Experimenty: přístupy

Generation-based Open-QA

- Alternativní přístup k Open QA – modelovat to jako úkol predikce sekvencí
  - jednoduše zakódovat otázku a dekódovat odpověď token za tokenem podle zakódování
- Ze začátku nebylo jasné jak zahrnout hodně znalostí do modelu
  - GPT-2 naznačil možnost přímého generování odpovědí bez kontextu pomocí sequence-to-sequence
    - to ale nebylo doladěno, pač to nestihli
  - T5 model ukázal že generovat přímo odpovědi bez explicitní extrakce kontextu je nadějný přístup
    - bohužel experimentovali jen na reading comprehension úkolech, kde byl k dispozici i kontextový dokument

REALM tedy finetunovali na open QA a porovnává se s base, large a and extra large modely T5

Výsledky


IDEA
RESEARCH
STRATEGY
SUCCESS!
PLANNING
RESULTS
TACTICS
EXECUTION

Name	Architectures	Pre-training	NQ (79k/4k)	WQ (3k/2k)	CT (1k /1k)	# params
BERT-Baseline (Lee et al., 2019)	Sparse Retr.+Transformer	BERT	26.5	17.7	21.3	110m
T5 (base) (Roberts et al., 2020)	Transformer Seq2Seq	T5 (Multitask)	27.0	29.1	-	223m
T5 (large) (Roberts et al., 2020)	Transformer Seq2Seq	T5 (Multitask)	29.8	32.2	-	738m
T5 (11b) (Roberts et al., 2020)	Transformer Seq2Seq	T5 (Multitask)	34.5	37.4	-	11318m
DrQA (Chen et al., 2017)	Sparse Retr.+DocReader	N/A	-	20.7	25.7	34m
HardEM (Min et al., 2019a)	Sparse Retr.+Transformer	BERT	28.1	-	-	110m
GraphRetriever (Min et al., 2019b)	GraphRetriever+Transformer	BERT	31.8	31.6	-	110m
PathRetriever (Asai et al., 2019)	PathRetriever+Transformer	MLM	32.6	-	-	110m
ORQA (Lee et al., 2019)	Dense Retr.+Transformer	ICT+BERT	33.3	36.4	30.1	330m
Ours ($\mathcal{X}$ = Wikipedia, $\mathcal{Z}$ = Wikipedia)	Dense Retr.+Transformer	REALM	39.2	40.2	**46.8**	330m
Ours ($\mathcal{X}$ = CC-News, $\mathcal{Z}$ = Wikipedia)	Dense Retr.+Transformer	REALM	**40.4**	**40.7**	42.9	330m

Tabulka výsledků na Open QA testech

- ukazuje accuracy různých přístupů na 3 open QA datasetech a REALM je významně překonává
- Navíc ukazuje parametry každého modelu
- Predikce ohodnoceny pomocí EM skóre s referenční odpovědí
- REALM významně překonává všechny předchozí modely
- ORQA je taky dobrá, ale REALM je lepší asi díky lepší pretraining metodám
- T5 (11b) je mocný a překonal předchozí modely, ale za cenu velkého počítacího nárůstu (50x větší od base)

Analýza

Výsledky pro NaturalQuestions-Open po opravení kritických komponent REALmu

Zároveň report jak často se gold answer objevila v top 5 retrievalech před finetuningem

Encoder or retriever

- Cíl byl zjistit zda pretraining realm zlepšuje retriever nebo encoder
- Proto resetovali params na jejich baseline před předtrénováním podle ORQA
- Zjistili, že encoder i retriever se zlepšují z trénování REALMU nezávisle, ale nejlepších výsledků dosahují při spolupráci table 2

Maskovací schéma

- Porovnávají jejích salient span masking s random token masking z BERTA a s random span masking z SpanBERTa
- Pro REALM je salient span mask kruciální pro BERTA ne

Příklady retrieved dokumentů

Příklad experimentů na NaturalQuestion-Open datasetu po odstranění kritických komponent REALMu

Navíc reportují, jak často se gold answer objevovala v top 5 dokumentech před finetuningem

Table 2. Ablation experiments on NQ's development set.

Ablation	Exact Match	Zero-shot Retrieval Recall@5
REALM	38.2	38.5
REALM retriever+Baseline encoder	37.4	38.5
Baseline retriever+REALM encoder	35.3	13.9
Baseline (ORQA)	31.3	13.9
REALM with random uniform masks	32.3	24.2
REALM with random span masks	35.3	26.1
30× stale MIPS	28.7	15.1

- Během pretrainu navíc proveden paralelní proces znovu vkládání dokumentů a znovu sestavování indexu MIPS
- Výsledkem je 1 obnovení indexu za cca 500 kroků tréninku
- K ukázání důležitosti častých aktualizací indexů je porovnali s obnovovací frekvencí.
- Výsledky v tabulce 2 naznačují, že zastaralý index může poškodit train modelů.

Příklad REALM MLM predikcí

- “Fermat” je správné slovo
- REALM (c) mu dává mnohem větší pravd. než BERT (a)
- REALM (b) dokáže načíst odpovídající dokumenty, ještě víc se tím zvyšuje pravd. správné odpovědi
  - to ukazuje, že REALM je schopen načíst dokument k vyplnění maskovaného slova, i když je trénován pouze na unlabeled textu

REALM: Retrieval-Augmented Language Model Pre-Training

Table 3. An example where REALM utilizes retrieved documents to better predict masked tokens. It assigns much higher probability (0.129) to the correct term, “Fermat”, compared to BERT. (Note that the blank corresponds to 3 BERT wordpieces.)

x:	An equilateral triangle is easily constructed using a straightedge and compass, because 3 is a ___ prime.		
(a) BERT	$p(y = \text{“Fermat”} \mid x)$	$= 1.1 \times 10^{-14}$	(No retrieval.)
(b) REALM	$p(y = \text{“Fermat”} \mid x, z)$	$= 1.0$	(Conditional probability with document z =“257 is … a Fermat prime. Thus a regular polygon with 257 sides is constructible with compass …”)
(c) REALM	$p(y = \text{“Fermat”} \mid x)$	$= 0.129$	(Marginal probability, marginalizing over top 8 retrieved documents.)

Budoucí práce

Prezentovaná práce je minimálním doloženým příkladem z celé rodiny přístupů založených ne REALM, kde je reprezentace předtrénovaná na velkém neolabelovém korpu pomocí dedukcí znalostí za běhu

Cílem je zobecnit tuto práci na strukturované znalosti, což by vedlo k

- zobecnění získávání kontextových souvislostí a jazykových modelů
- Vícejazyčnému nastavení
  - například při získávání znalostí v jazyce s hodně daty na reprezentaci znalostí v jazyce s méně daty
- multimodálnímu nastavení
  - například načítání obrázků či videí co mohou poskytnout znalosti zřídka získatelné z textu

Zdroje

Hlavní článek

- REALM (https://arxiv.org/pdf/2002.08909.pdf)
  - Výhody: článek je dobře srozumitelný a čitelný, obsahuje hezké obrázky popisu modelu
  - Nevýhody: -

Další články:

- BERT (https://arxiv.org/pdf/1810.04805.pdf)
  - Výhody: článek dobře popisuje model BERT, řešené problémy a výsledky
  - Nevýhody: popis modelu je třeba si místy přečíst několikrát, aby ho čtenář dobře pochopil
- T5 (https://arxiv.org/pdf/1910.10683.pdf)
  - Výhody: pěkné diagramy popisující fungování modelu
  - Nevýhody: článek je velice dlouhý a špatně se v něm hledají nějaké detaily, které by také mohly být lépe a stručněji vysvětlené